VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Aplikace pro správu pronájmu budov a přilehlých ploch

Application for Administration of Lease of Buildings and Adjacent Premises

2017 Dominik Voda

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Dominik Voda**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Aplikace pro správu pronájmu budov a přilehlých ploch
Application for Administration of Lease of Buildings and Adjacent Premises

Jazyk vypracování: čeština

Zásady pro vypracování:

V současné době existuje ve správě obcí mnoho objektů a prostor, jež bývají převzaty po jiných organizacích, pro které je hledáno využití, např. prostřednictvím pronájmu. Stále ještě často používaný způsob správy těchto budov a prostor a evidence nájmů vedená v papírové podobě není již dostačující a vyhovující současným požadavkům.

Cílem práce bude vytvoření aplikace pro správu pronájmu budov a prostor v objektech tohoto typu. Bude se jednat o aplikaci využívající webových technologií včetně nastavení zabezpečení v rámci rolí pro přístup z jiných systémů a kompetentním osobám. Aplikace bude obsahovat algoritmy a postupy pro 2D vizualizaci a správu nájmů v těchto budovách a prostorách k nim přilehlých v rámci daného objektu.

Jednotlivé body bakalářské práce:
1. Návrh aplikace na základě požadavků zadavatele.
2. Návrh datových struktur vhodných pro správu nájmů a 2D vizualizaci budov a přilehlých ploch v objektu .
3. Implementace aplikace.
4. Uživatelské testování aplikace a zhodnocení výsledků.
5. Návrhy na další rozšíření architektury aplikace.

Seznam doporučené odborné literatury:

[1] Symphony documentation. URL: http://symfony.com/doc/current/index.html
[2] PHP Manual. URL: http://php.net/manual/en/
[3] HTML5 Introduction. URL: http://www.w3schools.com/html/html5_intro.asp

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Kateřina Slaninová, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28. dubna 2017 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 28. dubna 2017 .....

Rád bych na tomto místě poděkoval Ing. Kateřině Slaninové, Ph.D. za odborné vedení, za pomoc a rady při zpracování této práce.

Abstrakt

Tato práce popisuje informační systém pro správu pronajatých budov a přilehlých objektů pro obec Velká Polom, který byl vytvořen v rámci této bakalářské práce. Obsahuje informace o tom, jakým způsobem probíhala prvotní analýza problémů a její výstupy, dále pak popis jednotlivých využitých technologií jak pro business logiku aplikace, tak pro uživatelské rozhraní. V rámci uživatelského rozhraní se zabývá implementací 2D vizualizace objektů, včetně zdůvodnění použití technologie SVG. Je zde také popsáno, jakým způsobem probíhalo testování jak na straně vývoje, tak na straně klienta, a protože se jedná o první fázi projektu, část práce se také věnuje funckionalitám, které by bylo možné do budoucna doimplementovat.

Klíčová slova: informační systém, databáze, 2D vizualizace, web

Abstract

This work describes an information system for administration of leased buildings and adjacent buildings for the municipality of Velká Polom, which was created in the scope of this bachelor thesis. It contains information on how the initial problem analysis and its outputs was carried out, as well as the description of each used technology for both the business logic of the application and the user interface. Within the user interface, it deals with the implementation of 2D object visualization, including the rationale for the use of SVG technology. It also describes how testing was performed both on the development side and on the client side, and since this is the first phase of the project, part of the work also deals with functionalities that could be implemented in the future.

Keywords: information system, database, 2D visualisation, web

Použitý seznam zkratek a symbolů

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ARES	Administrativní registr ekonomických subjektů
CSS	Cascading style sheet
DOM	Document object model
DIČ	Daňové identifikační číslo
HTML	Hypertext markup language
IČ	Identifikační číslo
JS	JavaScript
JSON	JavaScript object notation
MVC	Model View Controller
ORM	Object relation mapper
REST	Representational state transfer
SMTP	Simple mail transfer protocol
SVG	Scalable vector graphics
UML	Unified modeling language
URL	Uniform resource locator
WYSIWYG	Write you see is what you get
XML	Extensible markup language
YAML	Ain't Markup Language

Seznam obrázků

Obsah

Úvod

Cílem této bakalářské práce je vyvinout webovou aplikaci pro obec Velká Polom, která umožní jednodušší a přehlednější práci s procesem pronajímání budov a přilehlých objektů, které obec Velká Polom vlastní. Důležitou částí aplikace je tedy přehledný proces umožňující spravovat nájemní smlouvy a jejich dodatky včetně informací o jednotlivých nájemnících.
Druhou důležitou částí aplikace je potom přehledný způsob vizualizace jednotlivých objektů, umožňující rozlišit míru zaplnění jednotlivých objektů napříč celým systémem. Jde tedy o vizualizace jak na úrovni celého areálu včetně přilehlých pozemků a budov, tak na úrovni místností v jednotlivých budovách, resp. jejich patrech, včetně speciálního způsobu vizualizace budov typu „Hangár“.
Kapitola 1 se věnuje popisu současného stavu správy nájmů. Druhá kapitola potom popisuje požadavky a analýzu systému, včetně návrhu datového modelu. Následující dvě kapitoly obsahují popis uživatelského rozhraní a 2D vizualizace, společně s popisem jednotlivých technologií. Kapitola číslo 5 obsahuje informace o použitých technologiích v logické vrstvě aplikace a zá ní následuje předposlední část, která popisuje průběh testování celého projektu. V poslední kapitole jsou pak popsána možná vylepšení systému do budoucna.

1 Současný stav správy pronajatých pozemků a budov

Obec Velká Polom má ve své správě několik objektů převzatých od jiných organizací, pro které hledá využití, v současné době nejčastěji pronájem. Nyní jsou veškeré informace o nájemcích a jejich smlouvách v papírové podobě uloženy přímo na úřadě. Pro přehled využití jednotlivých objektů používají tabulku v MS Excel. Současně chybí jakákoli evidence jednotlivých nájemců, veškerá data jsou uložena v jednotlivých smlouvách a dodatcích.

U nájemce se eviduje jeho obchodní jméno, IČ případně DIČ, adresa, bankovní spojení a kontakt v podobě telefonního čísla a e-mailové adresy. V každé smlouvě je nutné jednoznačně identifikovat na koho se smlouva váže. Většinu objektů si pronajímají firmy, ty jsou jednoznačně identifikovány IČ, pro fyzické osoby se pak nejčastěji využívá jejich rodné číslo.

Když se objeví nový zájemce o pronájem, je potřeba aby pracovník obce prošel tyto tabulky, a dokumenty, aby zjistil, co může potenciálnímu nájemci nabídnout a za jakých podmínek.

Tento způsob správy je již nedostačující a nevyhovuje současným požadavkům, proto je nutné vyvinout systém, který umožní efektivně vizualizovat tato data a jejich správu.

1.1 Typy pronajímaných objektů

V areálu se nachází několik budov a pozemků. Budovy se dále dělí na patra a v nich jsou jednotlivé místnosti. Je možno si pronajmout jak jednotlivé místnosti, tak i pozemky, které se v areálu nacházejí.

Existuje několik budou typu „hangár“, se kterými se v systému pracuje jinak než s klasickou budovou. Hangáry jsou velké haly, které nejsou fyzicky rozděleny na místnosti jako u klasických budov, ale využívají systém kójí. Jejich velikost je možné přizpůsobit požadavkům nájemce. Nájemce si v hangáru často pronajme několik takových kójí, které na sebe navazují, a tak může vzniknout různě velký prostor, který je pak popsán ve smlouvě o nájmu.

1.1.1 Místnost

Místnost je nejčastěji se vyskytující pronajímaný objekt. Cena za pronájem místnosti je dána její velikostí a službami, které jsou v dané místnosti poskytovány. Místnosti jsou umístěny v patrech, patra jsou umístěna v budovách.

1.1.2 Pozemek

S pozemkem se pracuje podobně jako s místností, nejsou však vázány na patra či budovy. Pozemky se v areálu nacházejí poblíž jednotlivých budov.

1.2 Smlouvy a dodatky

Smlouvy mezi obcí a nájemci jsou jedinými dokumenty popisujícími, jaké objekty jsou aktuálně pronajaty. Smlouva mimo jiné obsahuje soupis pronajímaných objektů, včetně služeb, které nájemce pro daný objekt požaduje.

Velká Polom používá dva typy smluv:

- Na dobu určitou, kde je jasně definovaná platnost smlouvy datem. Pokud chce nájemce pronajímat objekty nadále, je nutné po vypršení takové smlouvy podepsat smlouvu novou
- Na dobu neurčitou, smlouva na začátku nemá jasně dané datum vypršení.

Aby bylo možno smlouvy v době jejich platnosti upravovat, je využíváno dodatků ke smlouvě. Tento dodatek zaznamenává nové nastavení služeb k daným objektům, případně je možné skrz dodatek úplně změnit, které objekty daný nájemce pronajímá. Obsahuje tedy soupis pronajímaných objektů včetně služeb, které jsou na ně vázány. Dodatek je vázán na původní smlouvy, smlouva je tedy upravována jejím posledním uzavřeným dodatkem.

2 Požadavky a návrh aplikace

Základním požadavkem je vytvořit systém, který jejím uživatelům umožní jednoduše spravovat nájemní smlouvy a dokáže přehledně informovat o vytíženosti jednotlivých budov případně jejich pater. Uživatel systému bude moci na základě požadavků potenciálního nájemce rychle a efektivně dohledat objekt, který dané požadavky splňuje, tzn. nabídne k pronájmu ty objekty, které se nejvíce blíží požadavkům klienta.

2.1 Objekty aplikace

Systém se skládá z několika objektů, v této kapitole jsou popsány ty nejdůležitější z nich včetně jejich vazeb. V původním zadání bylo také řešeno dynamické zadávání rozměrů místností, tento požadavek byl pak v průběhu vývoje upraven. Objektový model však s touto variantou počítal, a proto obsahuje tabulky place a ground, které slouží k ukládání dat abstraktních entit Place a Ground, které pak sjednocují chování těchto objektů v systému, ať se jedná o fixně zadané rozměry objektů nebo dynamicky definované.

2.1.1 Nájemce

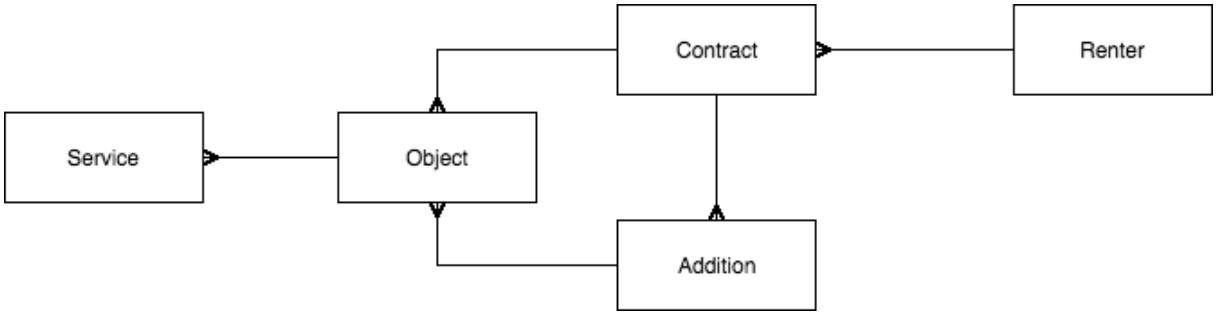
Nájemce může být jak fyzická, tak právnická osoba. Nájemce je ten, který s obcí podepisuje smlouvu o pronájmu. V systému může jeden nájemce mít několik smluv.

2.1.2 Smlouva

Smlouva je objekt, který propojuje nájemce a jeho pronajímané objekty. Nájemce může ve smlouvě pronajímat několik objektů s různými typy služeb. Smlouvy mohou být dále rozšířeny pomocí dodatků, které upravují, jaké objekty a služby na ně vázané budou nově pronajímány.

2.1.3 Dodatek

Při vytvoření smlouvy vzniká tzv. nultý dodatek, který obsahuje informace, které objekty a jejich služby daná smlouva obsahuje. Lze vytvářet i další dodatky, které na sebe navazují a v rámci jedné smlouvy tak upravují její strukturu. Smlouva je pak definována posledním platným dodatkem, který se k ní váže. Na Obrázku 1 jsou tyto vazby zjednodušeně zobrazeny.



Obrázek 1 - Zjednodušený diagram vazeb objektů

2.1.4 Pronajímaný objekt

Pronajímaný objekt je nejpodstatnějším předmětem smlouvy/dodatku, v systému se vyskytují dva typy objektů, místnosti a pozemky. Každý objekt má své unikátní označení, definovanou výměru pomocí šířky a délky a také koordináty x a y, které slouží pro jeho následnou vizualizaci v rámci areálu, popřípadě u místnosti v rámci jejího patra.

2.1.4.1 Místnost

V systému jsou dva typy místností:

- **Standardní**: místnost definována šířkou a délkou v metrech, je vázána na patro, patro je vázáno na budovu.
- **Hangárová**: speciální typ místnosti, mají předpoklady k tomu, aby mohlo být pronajato více takových místností zároveň. Je také vázaná na patro, a to je vázáno na budovu typu „hangár“. Tyto místnosti mají speciální pravidla pro pojmenovávání, ze kterých se pak vychází při manipulaci v systému.

Místnosti jsou vizualizovány pozicí pomocí koordinátorů x a y, které určují její pozici vzhledem k danému patru, respektive budově.

2.1.4.2 Pozemek

S pozemkem se v systému pracuje podobně jako se standardní místností, nemá ovšem žádné vazby na patra či budovy. Je jen jeden typ pozemků a ty jsou definovány šířkou a délkou v metrech a jeho pozice potom koordináty x a y.

2.1.4.3 Služba

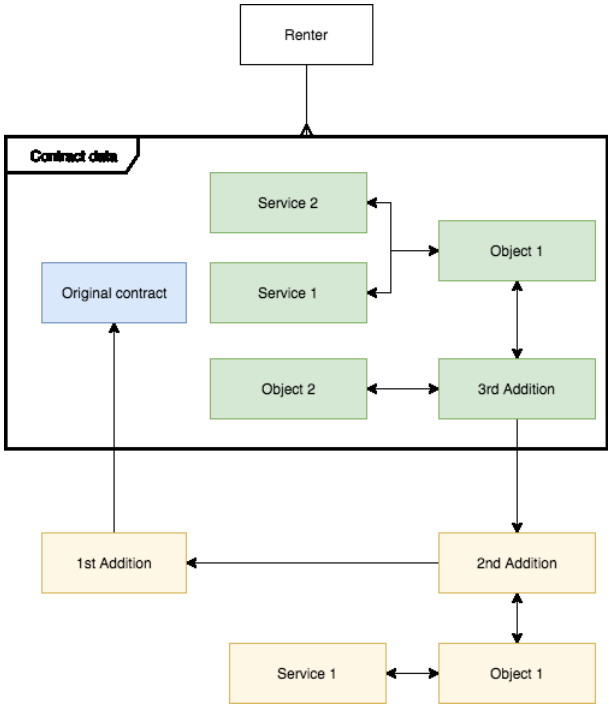
V rámci smlouvy jsou pro jednotlivé objekty, které si nájemce pronajímá také uvedeny, které služby jsou s nájmem daného objektu spojené. Může se jednat například o vodu, elektřinu, úklid atd. U služby se uvádí typ, v jakých intervalech je placena (měsíčně, ročně), o jaký typ platby jde (paušální, fixní) a kolik daná služba stojí.

2.1.4.4 Uživatel

Uživatel je reálný člověk identifikovaný uživatelským účtem, přes který je možné se do systému přihlásit a pracovat v něm. Uživatel je identifikován e-mailem a heslem.

2.2 Správa smluv

Základním prvkem informačního systému jsou smlouvy, smlouva je prvek, který propojuje nájemce s pronajímanými objekty a službami na ně vázanými. Smlouva nabývá platnosti určením data platnosti a končí nastaveným datem ukončení. O samotné propojení mezi smlouvou a pronajímanými objekty se starají dodatky, vytvořením nové smlouvy vzniká tzv. nultý dodatek, který obsahuje informace o objektech, které si nájemce pronajímá v rámci dané smlouvy. Aby bylo možné smlouvu během doby její platnosti měnit a upravovat, je možné vytvářet další dodatky, které původní dodatek zneplatní. Dodatek je vázaný na původní smlouvu, případně, pokud už daná smlouva dodatek obsahovala, váže se na poslední platný dodatek. Výsledná vazba mezi nájemcem a jeho pronajímanými objekty je tedy dána posledním dodatkem smlouvy. Schéma této struktury lze vidět na Obrázku 2.



Obrázek 2 - UML struktury smlouvy

2.3 Use cases

Na základě prvotní analýzy vyplynulo několik use casů. V této podkapitole popisuji některé use casy, které jsou v rámci aplikace řešeny. Use casy vychází z práce Bc. Radima Moravce, který tento projekt zpracovával v rámci své diplomové práce.

2.3.1 Vytvoření nájemce

Tento use case popisuje proces vytvoření nového nájemce, včetně možnosti předvyplnění dat formuláře pomocí ARES.

UC#001 Vytvoření nájemce
Popis: Uživatel může vytvořit nového nájemce
Aktéři: Přihlášený uživatel
Vstupní podmínky: Uživatel se nachází na kartě s tabulkou nájemců
Hlavní scénář: 1. Uživatel vyplní formulář na pravé straně obrazovky 2. Vyplněné údaje potvrdí kliknutím na tlačítko Vytvořit 3. Systém přesměruje na detail nového nájemce
Výstupní podmínky: 1. Uživatel zadal validní údaje 2. Systém uložil změny 3. Systém přesměroval na detail nově vytvořeného nájemce
Alternativní scénář 1: 1. Uživatel nevyplnil všechny povinné údaje 2. Systém nevytvoří nového nájemce 3. Systém zobrazí chybové hlášky u nesprávně vyplněných polí
Alternativní scénář 2: 1. Uživatel vyplní IČ nového nájemce 2. Uživatel klikne na tlačítko Načíst u pole s IČ 3. Systém se pokusí dohledat ostatní informace z ARESu 4. Pokud systém nalezne data v ARESu, předvyplní je do formuláře 5. Pokračujeme bodem 2 z hlavního scénáře

2.3.2 Upravit nájemce

Tento use case popisuje proces úpravy údajů již vytvořených nájemců.

UC#002 Upravit nájemce
Popis: Uživatel může upravit informace o nájemci
Aktéři: Přihlášený uživatel
Vstupní podmínky: Uživatel se nachází na kartě s tabulkou nájemců
Hlavní scénář: 1. Uživatel klikne v tabulce na řádek s nájemcem, kterého chce upravit 2. Systém zobrazí detail vybraného nájemce 3. Uživatel klikne na tlačítko upravit nájemce 4. Systém zobrazí formulář pro úpravu informací o nájemci 5. Uživatel změny potvrdí kliknutím na tlačítko uložit
Výstupní podmínky: 1. Uživatel změnil údaje 2. Systém uložil změny

2.3.3 Vytvoření smlouvy

Tento use case popisuje proces, jakým způsobem probíhá vytváření nové smlouvy.

UC#003 Vytvoření smlouvy
Popis: Uživatel může vytvořit novou smlouvu
Aktéři: Přihlášený uživatel
Vstupní podmínky: 1. Uživatel se nachází na kartě s tabulkou smluv 2. V systému existuje nájemce pro novou smlouvu
Hlavní scénář: 1. Uživatel vyplní formulář na pravé straně obrazovky 2. Kliknutím na tlačítko pokračovat potvrdí zadané údaje 3. Systém jej přesměruje na detail nově vytvořené smlouvy
Výstupní podmínky: 1. Uživatel zadal validní údaje 2. Systém uložil změny a přesměroval na detail nové smlouvy
Alternativní scénář: 1. Uživatel zadal neplatné údaje 2. Uživatel kliknul na tlačítko pokračovat 3. Systém zobrazí chybovou hlášku u pole s nevalidními daty

2.4 Popis vazeb systému

Na základě analýzy vznikl návrh jednotlivých objektů a jejich vazeb. V této kapitole popíši strukturu jednotlivých databázových tabulek pro všechny hlavní objekty, které se v systému vyskytují, včetně diagramu, který vizuálně reprezentuje vazby mezi nimi.

2.4.1 Popis jednotlivých tabulek

V následující podkapitole přesněji popisuji databázové tabulky pro jednotlivé objekty aplikace, jejich strukturu a vazby pomocí cizích klíčů. U všech tabulek v systému je primární klíč pojmenován jako id a nachází se ve všech tabulkách mimo vazební tabulku služeb a objektů.

2.4.1.1 User

Kromě e-mailu a hesla pro identifikaci tabulka obsahuje také další informace o dané osobě (jméno a příjmení), dále pak informaci, v jaké roli tento uživatel vystupuje.

2.4.1.2 Contract

V tabulce contract jsou uložena všechna důležitá data o smlouvách, nachází se zde veškeré důležité časové údaje (datum podpisu a platnosti, popřípadě datum ukončení), ze kterých pak vyplývá, v jakém stavu se daná smlouva nachází. Dále pak unikátní číslo faktury a cizí klíč renter_id, který odkazuje na primární klíč tabulky s nájemci.

2.4.1.3 Renter

Tabulka uchovává veškerá data o nájemcích. Obchodní jméno, IČ, telefon, e-mail, adresu a bankovní spojení.

2.4.1.4 ServiceType

Tabulka služeb, které jsou v systému využívány. Obsahuje pouze primární klíč a název dané služby.

2.4.1.5 Addition

Tabulka dodatků obsahuje kromě unikátního čísla dodatku také cizí klíče previous_addition_id pro identifikaci předcházejícího dodatku, pokud obsahuje hodnotu NULL jedná se o nultý dodatek. Smlouva, na kterou se dodatek váže je identifikována cizím klíčem contract_id, active_addition_id identifikuje aktivní dodatek v seznamu všech dodatků smlouvy, pokud se active_addition_id shoduje s id daného dodatku, znamená to, že daný dodatek je aktivní.

2.4.1.6 ObjectEntity

Tato tabulka je v systému využita pro abstraktní entitu, která umožňuje využít různé typy pronajímaných objektů a slučuje tak jejich chování. Nese informaci o typu objektu, na který odkazuje pomocí cizích klíčů place_id pro místnosti nebo ground_id pro pozemky.

2.4.1.7 AdditionObject

Vazební tabulka pro spojení mezi objekty a smlouvami. Obsahuje pouze primární id a cizí klíče addition_id pro identifikaci dodatku a object_id pro identifikaci ObjectEntity.

2.4.1.8 Service

Tabulka služeb ukládá informace, které typy služeb jsou přiřazeny k daným AdditionObject, za jakou cenu, typ placení a interval placení. Typ služby je identifikován cizím klíčem service_type_id a AdditionObject pomocí addition_object_id.

2.4.1.9 Ground/FixedGround

Jak jsem psal v úvodu, objektový model byl navržen tak, aby umožnil pracovat s různými typy pozemků, jelikož je ale aktuálně používaný jen jeden typ, je možné tyto dva objekty brát jako jeden. Ground je skrze cizí klíč fixed_ground_id propojen s FixedGround, fixed ground navíc obsahuje atribut title, který slouží pro pojmenování pozemku.

2.4.1.10 Place/FixedPlace

Tyto dva objekty, podobně jako u pozemků, je možné v aktuální verzi považovat za jeden. Place je pomocí fixed_place_id propojen s FixedPlace, na rozdíl od FixedGround není FixedPlace konečnou entitou pro popis najímaného místa. Place obsahuje cizí klíč room_id propojující jej s tabulkou rooms.

2.4.1.11 Room

Mimo primární klíč id a atribut title sloužící k pojmenování dané místnosti, obsahuje tato tabulka také cizí klíč floor_id, který odkazuje do tabulky floor.

2.4.1.12 Floor

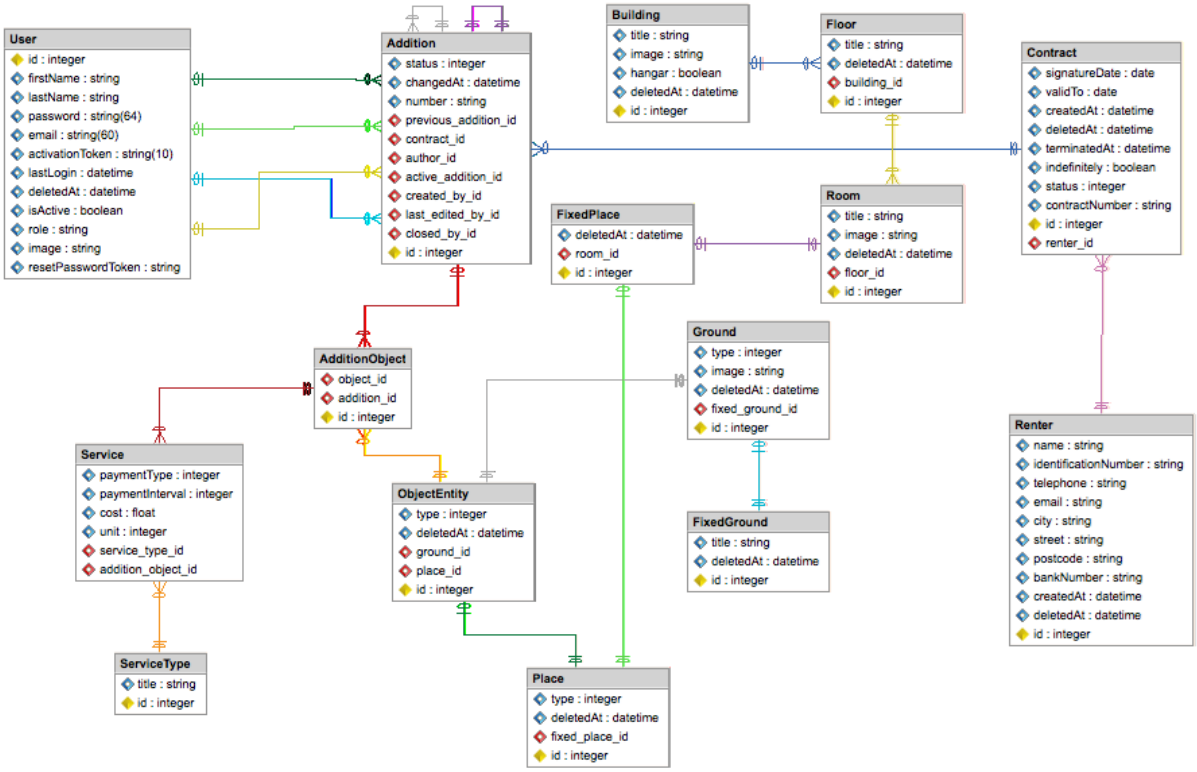
Tabulka, ve které jsou uloženy informace o patrech, kromě primárního klíče id navíc obsahuje cizí klíč building_id odkazující do tabulky building.

2.4.1.13 Building

Tabulka budov, mimo primární klíč také obsahuje boolean atribut hangar, kterým je pak dále v systému rozdělován typ místností v této budově na standardní a hangárové.

2.4.2 Entity relations diagram

Diagram na Obrázku 3 popisuje vazby mezi jednotlivými entitami systému. Pro každou takovou entitu existuje v databázi ekvivalentní tabulka, do které jsou ukládány stavy jednotlivých entit. Diagram byl navržen na základě předchozí analýzy.



Obrázek 3 - ER Diagram celého systému

2.5 Zabezpečení v rámci rolí

Po provedení analýzy a následné konzultace se zadavatelem se zjistilo, že v první fázi postačí pouze role přihlášený uživatel a nepřihlášený uživatel. Systém je však připraven na rozšíření o další role uživatelů a počítá tak s možností následné implementace nových rolí.

3 Uživatelské rozhraní

V této kapitole se věnuji popisu uživatelského rozhraní, jsou zde popsány jednotlivé technologie, které byly použity, včetně detailního popisu většiny částí aplikace.

Jelikož se jedná o webovou aplikaci, celé uživatelské rozhraní je psáno kombinací jazyků HTML5, JavaScriptu a CSS. Dodané uživatelské rozhraní je plně responsivní a je přizpůsobeno na plnou šířku zařízení.

Pro 2D vizualizaci objektů je pak použita knihovna D3, která přidává podporu pro vykreslování SVG objektů do stránky pomocí JavaScriptu.

3.1 Použité technologie

Na celém projektu bylo použito několik technologií sloužících jak pro samotné vykreslení uživatelského rozhraní, tak pro optimalizaci a minifikaci jednotlivých načítaných souborů. Využívá se také balíčkovací systém pro správu verzí jednotlivých knihoven a také systém umožňující automatizovat většinu úloh, které je potřebné provést při změnách v uživatelském rozhraní.

3.1.1 LESS

Less je jeden ze dvou aktuálně používaných jazyků pro preprocessing jazyka CSS umožňující upravovat styl jednotlivých HTML prvků. Less rozšiřuje původní CSS a přidává tak vývojáři nové možnosti jakým způsobem vytvářet konečný CSS soubor, tzv. „Syntactic sugar“. Přidává tak například možnost využívat proměnné, což se velmi hodí pro práci s barvami, dále pak umožňuje zapisovat zanořená pravidla pro selektory prvků, což výrazně napomáhá čitelnosti kódu a v konečném důsledku ho není potřeba napsat tolik. [1]

Využívání Less preprocesoru zvyšuje nejen udržitelnost a rozšiřitelnost kódu, ale umožňuje také jednoduše vytvářet různá témata vzhledu použitím jiné sady proměnných.

3.1.2 Bootstrap

Bootstrap je frontendový Framework umožňující rychle vytvářet uživatelská rozhraní, je vyvíjený jako open source a na tomto projektu je použit ve verzi 3. Framework se skládá z několika balíčků, které se dají použít i samostatně. Bootstrap umožňuje rychle vytvářet uživatelská rozhraní, protože obsahuje obecně používané prvky, které se opakují na většině webových stránek jako jsou: navigace, formuláře, seznamy, tabulky atd. Důležitou součástí je bootstrap grid.

Bootstrap grid je modul umožňující web rozdělit na řádky a sloupce, což výrazně urychlí vytvoření rozložení prvků na jednotlivých podstránkách a velmi usnadňuje vytvoření responsivního chování. Na tomto projektu je nastavený grid na výchozích 12 sloupců, toto nastavení je dostačující. [2]

3.1.3 jQuery a Typescript

Protože práce s čistým JavaScriptem přinaší mnoho nevýhod, je využíváno několika knihoven usnadňujících práci s různými částmi a aspekty JavaScriptu.

3.1.3.1 jQuery

JQuery je již dlouhou dobu nejrozšířenější JavaScriptový Framework, který značně usnadňuje vývojářům vyhledávat a manipulovat s DOMem.

Mezi nejčastější využívané funkcionality jQuery patří například vylepšená práce se selektory jednotlivých prvků, výrazné zjednodušení animování jednotlivých prvků na stránce, které je sice možno ve velké míře nahradit animacemi v CSS3, nicméně komplikovanější animace je potřeba stále napsat v JavaScriptu.

Samotnou kapitolou je potom práce s AJAXem, který je v tomto projektu využitý například k získání informací pro vykreslení SVG objektů, kde zdrojem pro tyto data je REST API. [3]

3.1.3.2 Typescript

JavaScript jako takový není objektově orientovaný jazyk, má sice podporu pro práci s objekty, ale na úrovni paměti pracuje pouze s prototypy a ty se pro některé využití jazyka nehodí. A jelikož JavaScript nemá ani žádnou ochranu datových typů a další věci, které mohou komplikovat vývoj větších aplikací, vznikl preprocessor Typescript.

Typescript přidává vývojářům možnost pracovat s JavaScriptem jako s jinými vyvinutějšími programovacími jazyky jako například C# nebo Java, dnes ale také PHP a další. To znamená že vývojářům umožňuje využívat třídy, jejich dědění včetně řízení přístupu k jednotlivým prvkům, přidává definice pro datové typy atd.

Výhodou využití tohoto preprocesoru je značné zlepšení rozšiřitelnosti, udržitelnosti ale také čitelnosti kódu, což jsou pro vývoj rozsáhlejších aplikací podstatné vlastnosti. [4]

3.1.4 Balíčkovací systémy a automatizované úlohy

Aby bylo možno spravovat verze a závislosti jednotlivých knihoven, existují tzv. balíčkovací systémy. Většina těchto balíčkovacích systému je založena na verzovacím systému GIT. Výhodou těchto systémů je možnost rychlého přepínaní verzí jednotlivých knihoven s čím souvisí i jejich aktualizace. V praxi to pak vypadá tak, že jsou do aplikace instalovány knihovny automaticky na základě konfiguračního souboru. V tomto projektu byly pro uživatelské rozhraní využity balíčkovací systémy NPM a Bower.

3.1.4.1 NPM

NPM je manažer balíčků pro aplikace v Node.js, byl vytvořen roku 2009 jako open source projekt, aby pomohl vývojářům vytvářet a sdílet JavaScriptové moduly. Jedná se tedy o veřejnou sbírku balíčků, které mohou vývojáři instalovat a publikovat pomocí příkazové řádky. [5]

3.1.4.2 Bower

Bower se svou funkčností velmi podobá NPM, zaměřuje se však na správu verzí knihoven sloužících pro vývoj uživatelských rozhraní jako takových. Jedná se tedy o kolekci komponent, ze kterých se pak dané uživatelské rozhraní skládá. Konkrétně v tomto projektu je skrze Bower nainstalován Bootstrap, WYSIWYG editor TinyMCE, ikonkový font – font-awesome aj.

3.1.4.3 Node.js

Jedná se o velmi výkonné, událostmi řízené prostředí pro JavaScript, jeho základem je javascriptový interpret V8 z Google Chrome s tenkou vrstvou kódu v C++ poskytující nutné rozšíření pro správné fungování. V tomto případě se využívá Node.js ve spojení s balíčkovacím systémem NPM a automatizovanými úlohami v Gruntu.

Node.js je serverový Framework, což nám umožňuje využívat JavaScript nejen na straně klienta, ale dokážeme jej použít i pro operace přímo na serveru nebo jako v našem případě při vývoji, jako nástroj pro kompilace a spojování CSS a JavaScript. [6]

3.1.4.4 Grunt

Grunt je nástroj pro správu automatizovaných javascriptových úloh. Nejčastěji se využívá pro minifikaci, kompilaci případně i unit testování a spojování souborů.

Minifikace je efektivní způsob, jak minimalizovat velikost CSS a javascriptových souborů na webu pro zvýšení výkonu. Obecně je to snaha odstranit z kódu zbytečné části, které prohlížeč nepotřebuje

k tomu, aby jej správně zpracoval. Jde například o komentáře, odřádkování a propracovanější skripty dokáží upravovat i syntaxi kódu tak, aby i při stejné funkčnosti zabíral méně místa.

Pro další zrychlení běhu aplikace se využívá spojování souborů, jde o snahu minimalizovat počet požadavků na server pro načtení jedné podstránky. Znamená to tedy, že i když se v aplikaci využívá několik javascriptových knihoven, je možné všechny jejich potřebné soubory sloučit do jednoho skriptu a teprve ten následně minifikovat.

Další automatizovanou úlohou je kompilace Less a TypeScript souborů.

Všechny tyto úlohy je možné jednoduše nakonfigurovat skrz tzv. Gruntfile, což není nic jiného než předpis pro jednotlivé úlohy. Jsme pak schopni z příkazové řádky vyvolat kteroukoli z těchto úloh a spustit ji ručně, ale je možné také využít file watcheru, který bude automaticky naslouchat na změny v námi definovaných souborech a na základě těchto změn spouštět potřebné úlohy automaticky. [7]

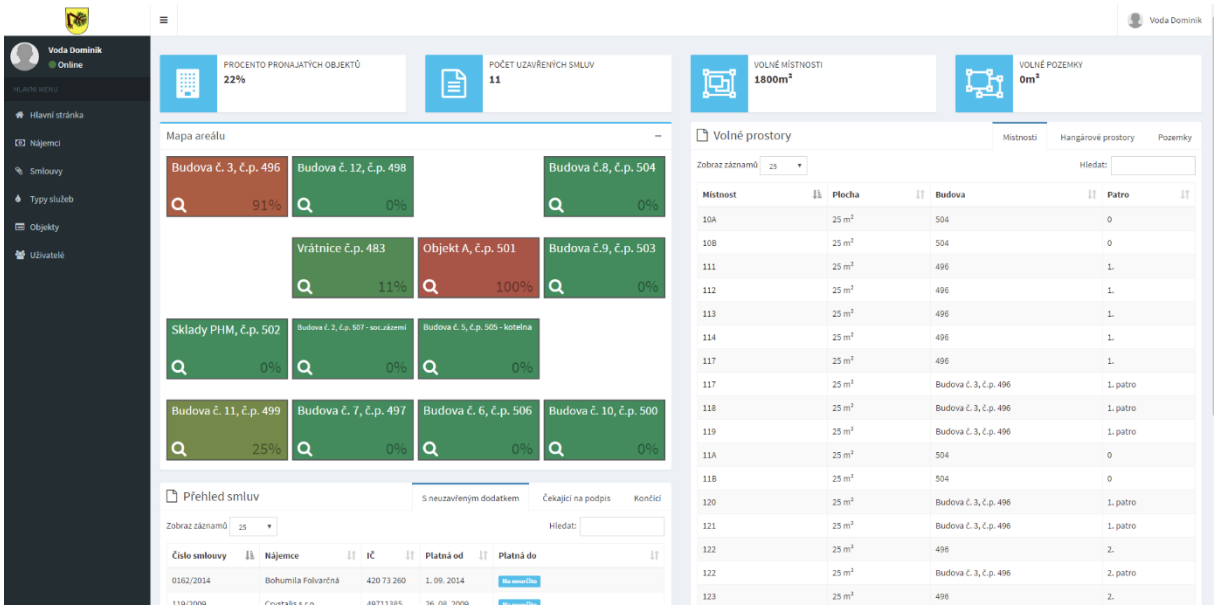
3.2 Popis uživatelského rozhraní

Uživatelské rozhraní je zabezpečená sekce, ve které je možné spravovat veškerá data, která jsou potřebná pro správné fungování systému. Pro přihlášení do systému je nutné mít zřízený uživatelský účet, na který se pak přihlásíte pomocí emailu a hesla. Přihlašovací formulář vidíme na Obrázku 4.

Obrázek 4 - Přihlašovací formulář

3.2.1 Úvodní obrazovka

Po přihlášení je uživatel automaticky přesměrován na úvodní obrazovku systému nazývanou jako „Hlavní stránka“. Náhled stránky je zobrazen na Obrázku 5.

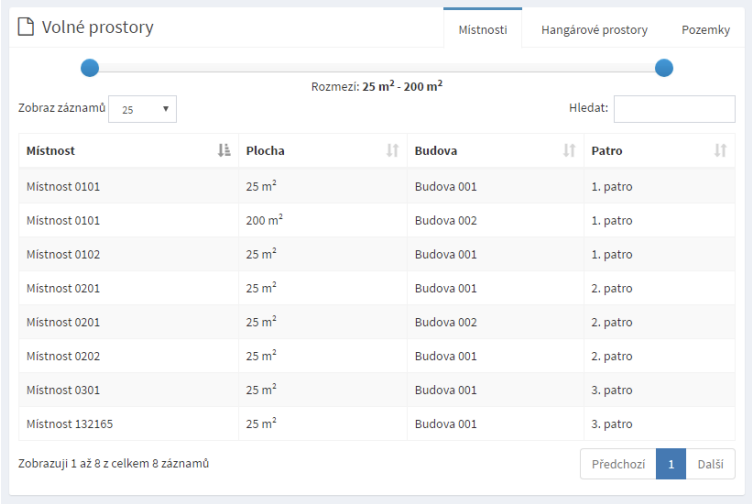


Obrázek 5 - Úvodní obrazovka

Hlavní stránka uživateli nabízí hrubý přehled o aktuálním stavu pronajímaných objektů a smluv, včetně možnosti rychlého vyhledávání mezi objekty, které je možné si pronajmout. Správce tak může rychle na mapě areálu vidět míru zaplnění jednotlivých budov pomocí 2D vizualizace.

3.2.1.1 Rychlé vyhledávání

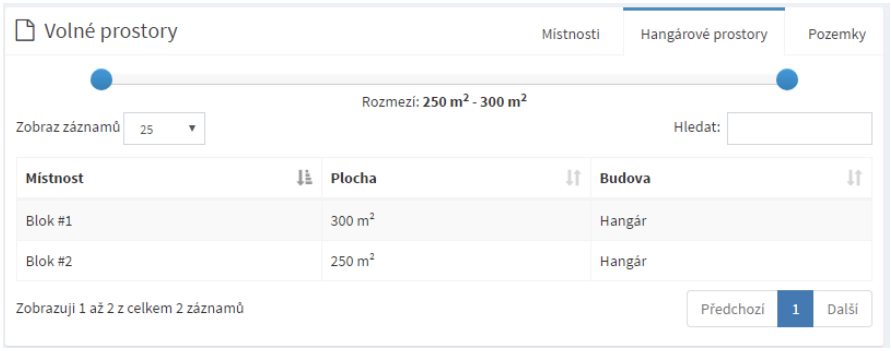
Jedním z bloků je také rychlé vyhledávání ve volných prostorách, správce tak může vyhledávat objekty buď pomocí fulltextového vyhledávacího pole nebo využít filtrační slider, který umožňuje nastavit rozmezí od a do pro velikost plochy, kterou chce uživatel vyhledat. Tato funkce může být velmi užitečná v momentě, kdy bude potřeba rychle zjistit, co je možné potenciálnímu nájemci nabídnout. Náhled tohoto boxu lze vidět na Obrázku 6.



Obrázek 6 - volné prostory (místnosti)

Tento vyhledávač pak zahrnuje i speciální funkčnost pro vyhledávání maximálního navazujícího prostoru v hangárových budovách. Hangár je speciální typ budovy, která má pouze jedno patro, v něm se však nachází velký prostor uměle rozdělený na několik kójí. Tyto kóje jsou rozmístěny ve dvou řadách (A a B), v každé řadě se nachází 12 takových kójí.

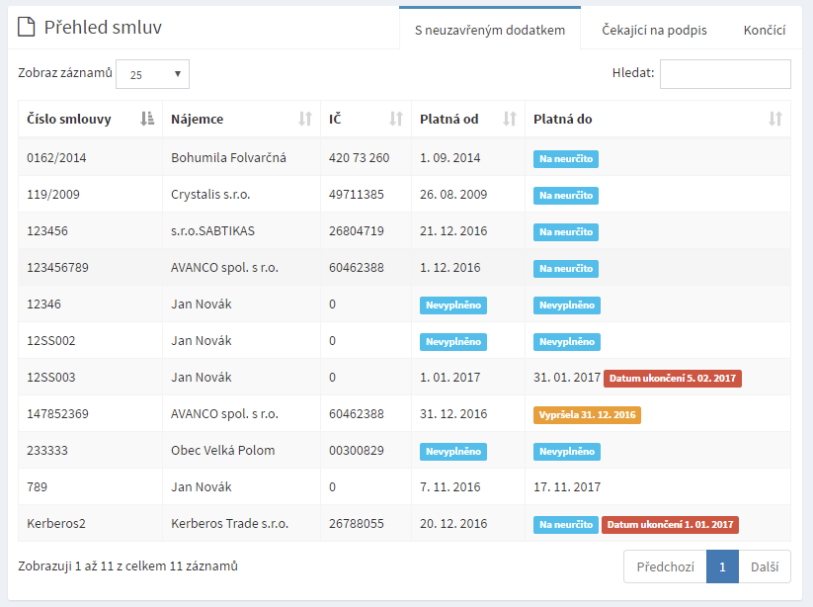
Nájemce však často potřebuje pronajmout více než jednu samostatnou kóji, proto vyhledávání v hangárových prostorách funguje tak, že se vždy vypočítají největší možné spojené prostory a tento spojený prostor se pak nabídne ve vyhledávání (viz Obrázek 7), tzn. když budou nepronajaté kóje 1A, 2A a 2B, každá o výměře 30 m^2, bude ve vyhledávání volný blok o výměře 90 m^2.



Obrázek 7 - Volné prostory (hangáry)

3.2.1.2 Rychlý přehled smluv

Aby uživatel rychle zjistil, které smlouvy jsou v nestandardním stavu, existuje na hlavní stránce blok s přehledem smluv (viz Obrázek 8). V tomto výpisu jde opět fulltextově vyhledávat.



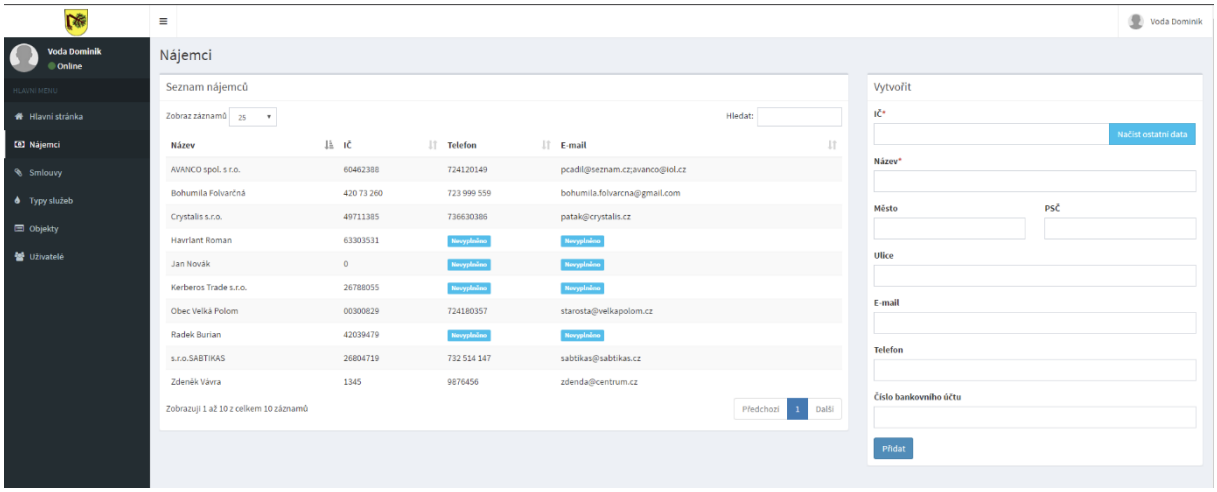
Obrázek 8 - Rychlý přehled smluv

Uživatel má přehled rozdělený do tří záložek:

- S neuzavřeným dodatkem: to jsou smlouvy, které mají dodatek, který zatím nebyl uzavřený, tzn. jedná se o novou smlouvu, u které ještě nebylo dokončeno sestavení jejich objektů nebo o již aktivní smlouvu, která má rozpracovaný nový dodatek.
- Čekající na podpis: smlouvy, které nemají stanovené datum podpisu, tzn. nemají určeno datum, od kterého jsou platné
- Končící: smlouvy na dobu určitou, kterým se blíží datum ukončení, popřípadě smlouvy na dobu neurčitou, u kterých bylo nastaveno datum ukončení.

3.2.2 Nájemci

Podstránka sloužící pro správu nájemců. Pro přehled je v levé části jednoduchá tabulka s vyhledáváním, v pravé části je pak formulář pro přidání nového nájemce. Náhled této stránky lze vidět na Obrázku 9.

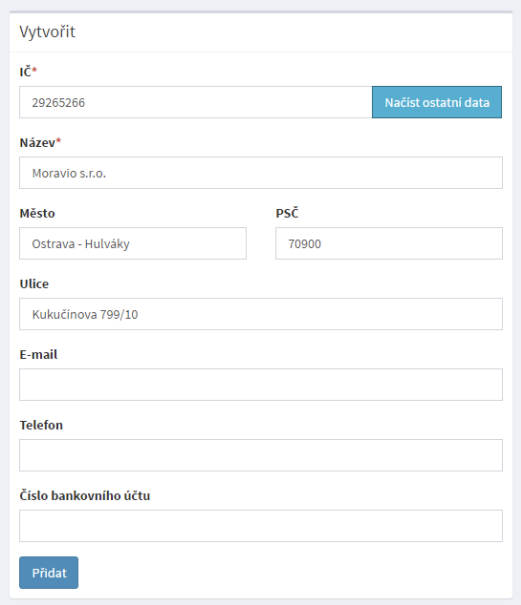


Obrázek 9 - Nájemci (přehled)

3.2.2.1 Formulář pro přidání + ARES

Zajímavou součástí formuláře pro vytvoření nového nájemce je našeptávač ARES – administrativní registr ekonomických subjektů. Jedná se o projekt ministerstva financí, který na jednom místě zpřístupňuje údaje z informačního systému pro vedení registrů a evidencí veřejné správy o ekonomických subjektech. Náhled tohoto formuláře lze vidět na obrázku 10.

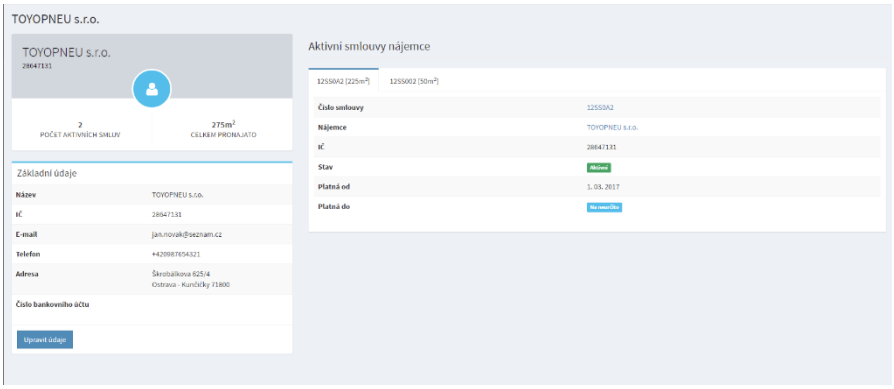
Tato webová aplikace nám umožní pomocí XML API vyhledat informace o ekonomickém subjektu na základě jeho identifikačního čísla (IČ) a pomocí získaných informací předvyplnit údaje ve formuláři.



Obrázek 10 - Přidání nájemce + ARES

3.2.2.2 Detail

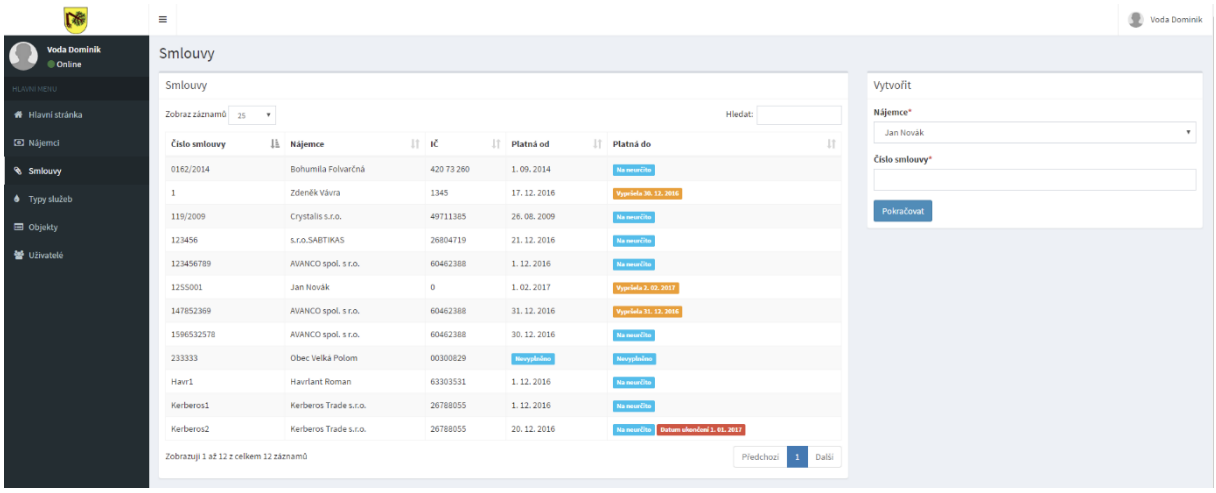
V detailu nájemce je možno upravovat jeho údaje, popřípadě vidíme přehled všech jeho aktivních smluv. Náhled této podstránky je zobrazen na obrázku 11.



Obrázek 11 - Detail nájemce

3.2.3 Smlouvy

Správa smluv je další důležitou částí systému, v přehledu je podobně jako u nájemců v levé části tabulka s výpisem jednotlivých smluv, v pravé pak formulář pro založení nové smlouvy.

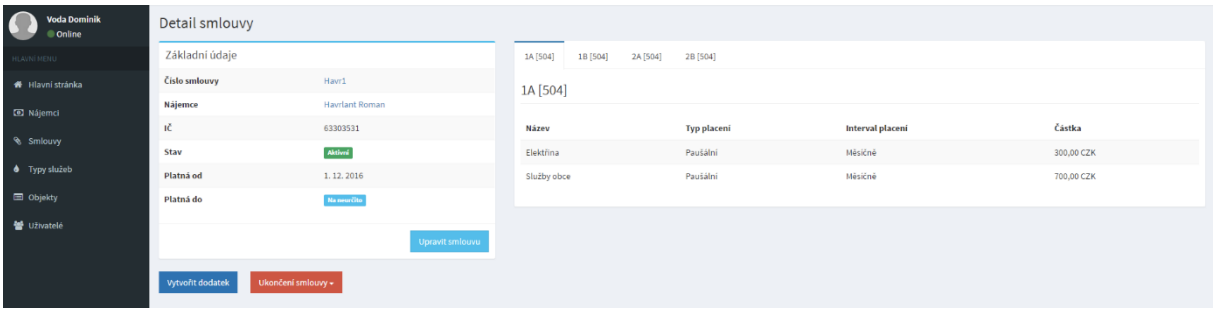


Obrázek 12 - Smlouvy (přehled)

Při vytváření nové smlouvy stačí vybrat k jakému nájemci bude patřit a vyplnit její číslo. Číslo smlouvy je unikátní a není možné zadat do systému dvě smlouvy se stejným číslem. Po kliknutí na tlačítko pokračovat, je uživatel přesměrován na detail nově vytvořené smlouvy. Náhled této podstránky lze vidět na Obrázku 12.

3.2.3.1 *Detail smlouvy*

V detailu smlouvy se nastavují veškerá data, která jsou pro smlouvu potřeba. Je zde vidět aktuální nastavení smlouvy rozdělené podle jednotlivých objektů, na které se smlouva vztahuje (viz Obrázek 13).

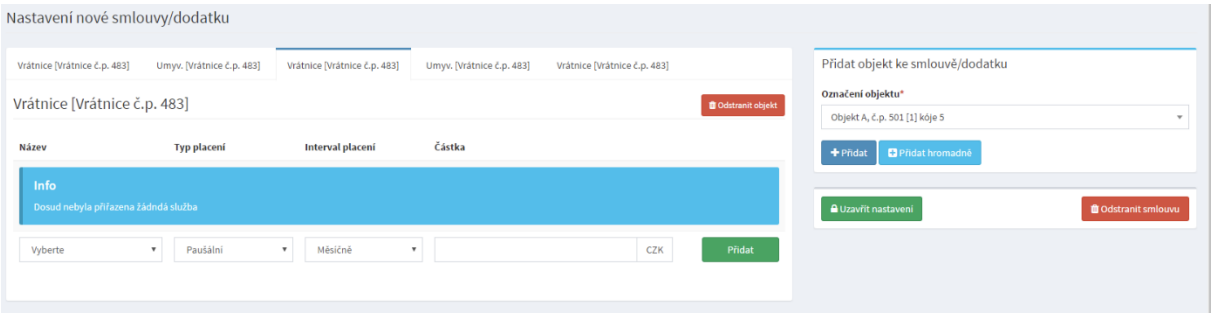


Obrázek 13 - Smlouvy (detail)

Pokud se jedná o smlouvu novou nebo takovou, která má rozpracovaný nový dodatek, nastavení, na které objekty se daná smlouva vztahuje, se děje zde.

3.2.3.2 *Nastavení nové smlouvy/dodatku*

Aby bylo možné přiřadit ke smlouvě objekty a na tyto objekty navázat jednotlivé služby, je ve spodní části obrazovky několik formulářů. Pokud smlouva/dodatek nemají přiřazený ještě žádný objekt, zobrazuje se pouze pravý formulář pro přiřazení objektu ke smlouvě. Uživatel v takovém případě z roletky vybere jeden z právě volných objektů a po kliknutí na tlačítko „Přidat“ se v levé části vytvoří záložka s daným objektem, ve které je pak možno pomocí spodního formuláře přiřazovat jednotlivé služby a jejich vlastnosti (viz Obrázek 14).



Obrázek 14 - Nastavení nové smlouvy/dodatku

3.2.3.3 *Přidat dodatek*

Pro úpravu nastavení již existující smlouvy je potřeba vytvořit dodatek. Po kliknutí na tlačítko „Vytvořit dodatek“ se ve spodní části obrazovky zobrazí stejné formuláře jako při prvotním nastavení nové smlouvy, ale protože se u dodatků často jedná o úpravy původního znění smlouvy a málokdy jde o kompletní změnu struktury smlouvy, systém automaticky načte nastavení původní smlouvy do nově vytvářeného dodatku. Ušetří tak práci s přidáváním původních objektů a jejich služeb, uživatel pak může jednoduše přenastavit objekty a služby, kterých se daná smlouva/dodatek týká.

3.2.3.4 Přidat hromadně

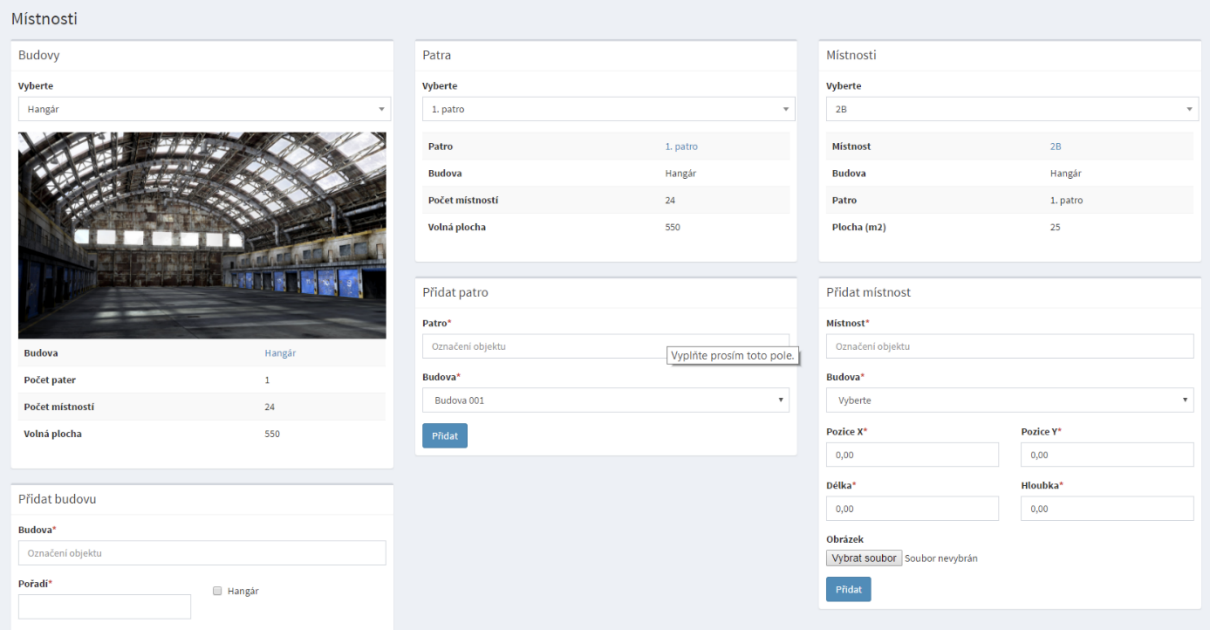
Pro usnadnění přidávání objektů do smlouvy, které mají stejné nastavení služeb, existuje funkce umožňující vybrat jednotlivé objekty a těm hromadně nastavit stejné služby a jejich vlastnosti. Po kliknutí na tlačítko „Přidat hromadně“ se otevře modální okno (viz Obrázek 15), ve kterém uživatel vybere, které objekty chce hromadně nastavit a poté navolí služby a jejich nastavení. Po uložení se pak takto nastavené objekty přenesou do levého panelu, kde je možné je pak upravovat jako klasické jednotlivé objekty.

Obrázek 15 - Hromadné přidání objektů

Tato funkce byla přidána zejména pro zadávání smluv, které souvisejí s pronájmem hangárových kójí, kde klasicky nájemce pronajímá několik kójí zároveň. Pro tyto kóje se pak ale zpravidla přiřazují stejné služby za stejné ceny.

3.2.4 Objekty

Pro správu jednotlivých objektů existuje několik podstránek. V menu je záložka „Objekty“ rozdělena na dvě skupiny: „Místnosti“ a „Pozemky“. Na stránce s místnostmi je pak přehled jednotlivých budov, jejich pater a místností, které se v těchto patrech nacházejí (viz Obrázek 16). Zde je také možnost jednotlivé objekty vytvářet, popřípadě kliknutím se dostat na jejich detail, kde je lze upravovat.

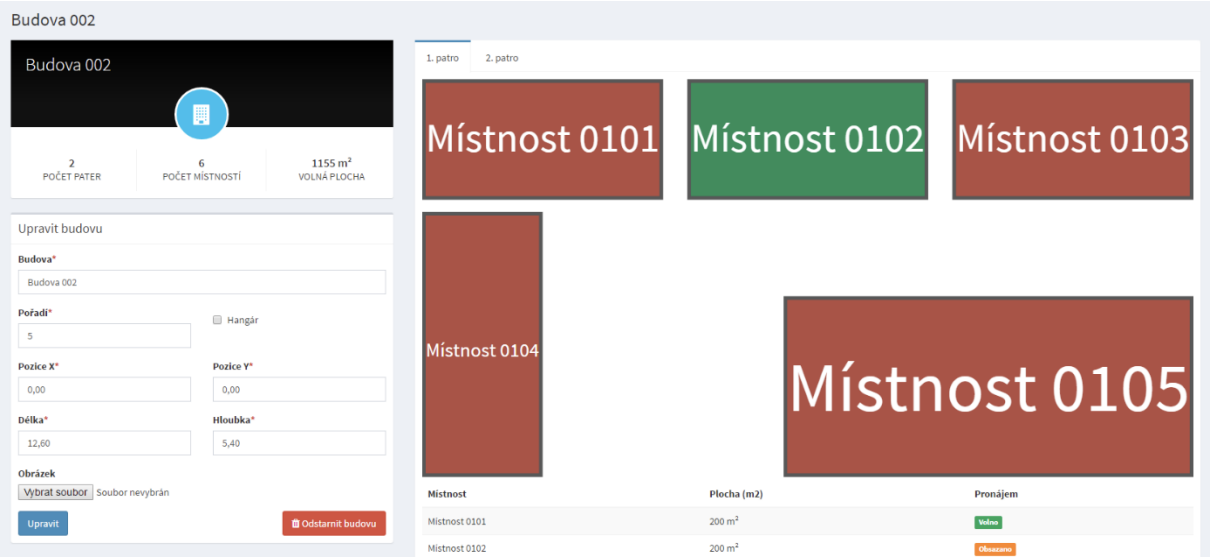


Obrázek 16 - Přehled objektů (místnosti)

Uživatel pak může jednoduše najít, jaké místnosti se v daných budovách nacházejí. V prvním kroku vybere budovu, kterou chce zobrazit. Po vybrání budovy se zpřístupní roletka pro výběr z pater, které se v dané budově nacházejí. Po vybrání patra se pak zobrazí roletka s přehledem místností tohoto patra.

3.2.4.1 *Detail budovy*

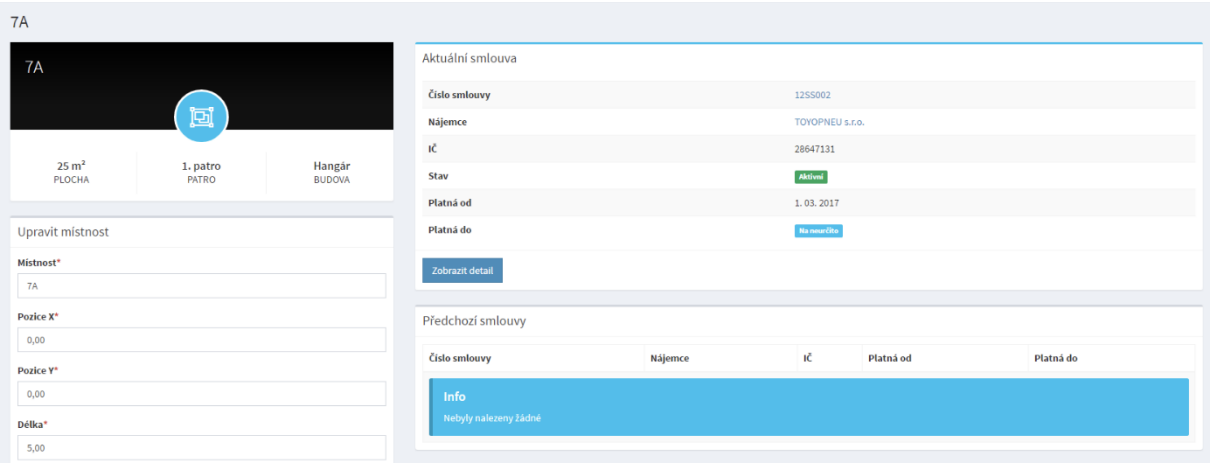
Detail budovy je mimo formulář pro úpravu rozšířený o vizualizaci jednotlivých pater. V pravé části se nachází panel se záložkami, každá záložka značí jedno patro. Pokud mají místnosti správně vyplněny koordináty x a y a nastaveny rozměry, jsou pak v této záložce zobrazeny vzhledem k danému patru. Barevně je pak rozlišeno, zda je daná místnost pronajata: zelená značí pronajato, červená pak že je místnost volná. Toto rozlišení lze vidět na Obrázku 17.



Obrázek 17 - Detail budovy

3.2.4.2 *Detail místnosti*

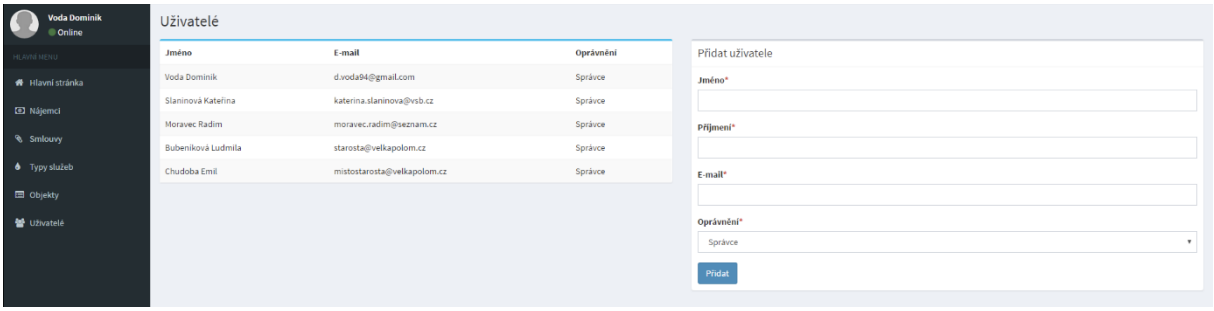
Detail místnosti je krom klasického formuláře pro úpravu údajů rozšířený o panel s aktuální smlouvou, ve které je daná místnost pronajímána. Pod tímto panelem se pak nachází seznam smluv, ve kterých byla pronajata v minulosti (viz Obrázek 18).



Obrázek 18 - Detail místnosti

3.2.5 Uživatelé

Správa uživatelů je tvořena dvěma jednoduchými obrazovkami. V přehledu je možné vidět seznam všech uživatelů s přístupem do systému, v pravé části se pak nachází formulář pro vytvoření nového uživatele. Náhled této podstránky lze vidět na Obrázku 19.



Obrázek 19 - Přehled uživatelů

4 2D vizualizace

Jedním z bodů zadání této práce bylo vytvoření 2D vizualizace spravovaných objektů. Vizualizovat by se mělo na dvou úrovních. První z nich je zobrazení na úrovni celého plánu areálu, tzn. jde o to vykreslit pozice budov a pozemků, které se v areálu nacházejí. Druhá úroveň je pak vykreslení místností daných budov, resp. jejich pater.

V průběhu vývoje přibyl ještě požadavek na vizualizaci speciálních budov – Hangárů.

4.1 Technologie

Jelikož se jedná o webovou aplikaci, bylo několik možností, jakým způsobem objekty vykreslovat.

4.1.1 Technologické možnosti

Níže popisuji nejpravděpodobnější technologie, které by bylo možné pro vykreslení použít, včetně jejich kladů a záporů.

4.1.1.1 CSS

Asi nejprimitivnějším způsobem by bylo vykreslování pomocí CSS stylů, rychle bychom ale narazili na omezení týkající se podpory napříč prohlížeči a také by se komplikovaně pracovalo se souřadnicovým systémem, který je pro vykreslení areálu nutný.

Plusy: jednoduché

Mínusy: špatná podpora práce s responzivním souřadnicovým systémem

4.1.1.2 Flash

Před deseti lety by byl Flash pravděpodobně volbou číslo jedna, dnes je již Flash na rychlém ústupu a většina moderních prohlížečů jej již ve výchozím nastavení nepodporuje z důvodu množství kritických bezpečnostních děr, které Flash obsahuje.

Plusy: mnoho možností práce s objekty a jejich interakce s uživatelem

Mínusy: upadající podpora, kritické bezpečnostní chyby

4.1.1.3 Canvas

Jedná se o technologií přicházející s nástupem HTML5. V DOMu je přidán element typu Canvas, což je vlastně plátno, na které se poté pomocí JavaScriptu vykreslují jednotlivé obrazce. [8, 9]

Plusy: dobrá podpora v prohlížečích, vhodné pro tento typ zadání

Mínusy: kreslit jde pouze JavaScriptem, jednotlivé obrazce neexistují v DOMu

4.1.1.4 SVG

Pomocí SVG lze jednoduše vykreslovat různé typy obrazců. V dnešní době je SVG standardem ve vykreslování vektorových obrazců na všech moderních webech. Důvodem je široká podpora napříč prohlížeči a zároveň není nutné SVG vykreslovat pouze JavaScriptem, ale o vykreslení prvku se stará přímo jádro prohlížeče. Je například hojně využíván pro vykreslování ikonek na webu, dále třeba log, které se dají převést do vektorových tvarů. Hlavním důvodem využívání SVG je snížení přeneseného objemu dat a souvisejícího zrychlení načítání webu.

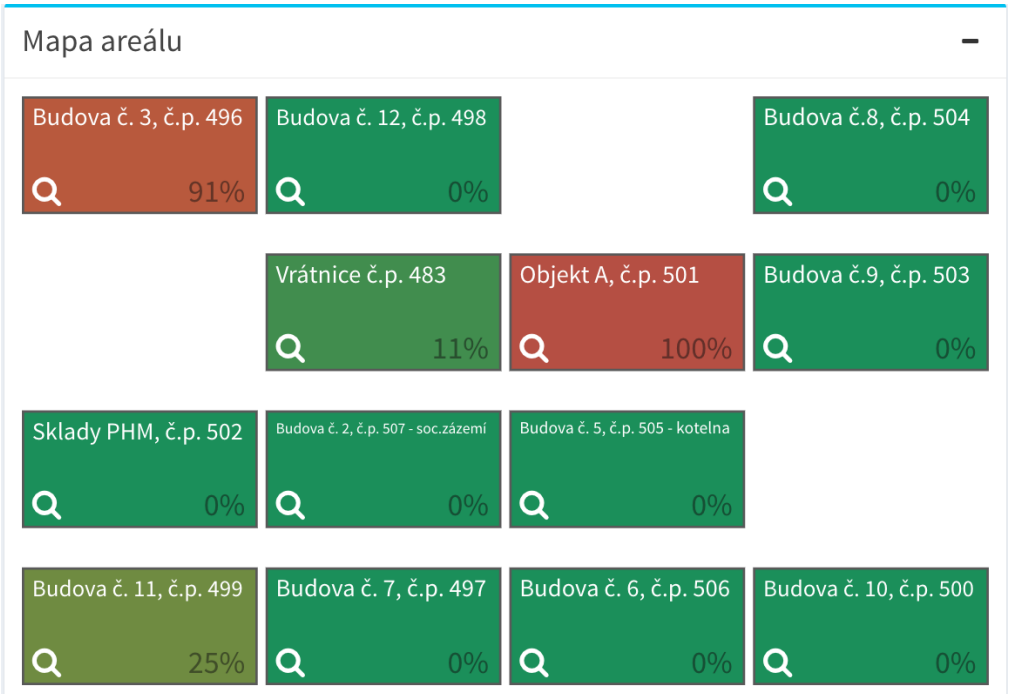
Plusy: objekty jsou součástí DOMu (lze je tak například stylovat pomocí CSS), možnost pracovat s responsivitou

Mínusy: v některých případech složitější zápis

4.2 SVG a vizualizace stavu objektů

Pro 2D vizualizaci jsem zvolil SVG, protože má dnes již dobrou podporu v prohlížečích, je možné jej upravovat i po načtení pomocí JavaScriptu a plátno, na které se objekty vykreslují, je možné přizpůsobit okolním rozměrům, což nám zanechá velkou míru responzivity celého webu.

Veškeré objekty jsou vykreslovány schematicky. Každá budova, pozemek či místnost má u sebe uloženy koordináty x a y a pro výměru parametry šířky a délky. Koordináty x a y slouží ke správnému umístění levého horního rohu vzhledem k okolí, šířka a délka pak slouží ke správnému vykreslení velikosti daného objektu. Tyto hodnoty jsou zadávány v metrech. Míra zaplnění jednotlivých objektů je znázorněna barevně, 0% pronajatých prostor je značeno červeně, červená pak postupně přechází do zelené, která značí 100% vytížení prostor a v takovém případě tedy platí, že jsou v budově všechny místnosti v daný okamžik pronajaty. Vykreslování pomocí SVG se využívá na dvou místech.



Obrázek 20 - SVG vizualizace celého areálu

Na hlavní stránce je pomocí SVG vykreslena mapa celého areálu, tzn. můžeme schematicky vidět znázornění jednotlivých budov, včetně míry zaplnění pronajímaných místností, které se v dané budově nacházejí (viz Obrázek 20). Vedle budov jsou také vykreslovány i veškeré pozemky.

V detailu jednotlivých budov pak můžeme vidět znázornění jednotlivých pater, kde jsou pomocí SVG vykresleny místnosti v daném patře. Koordináty x a y pro místnosti jsou zadávány vzhledem k patru, kde se místnost nachází.

4.2.1 ViewBox a používané prvky SVG

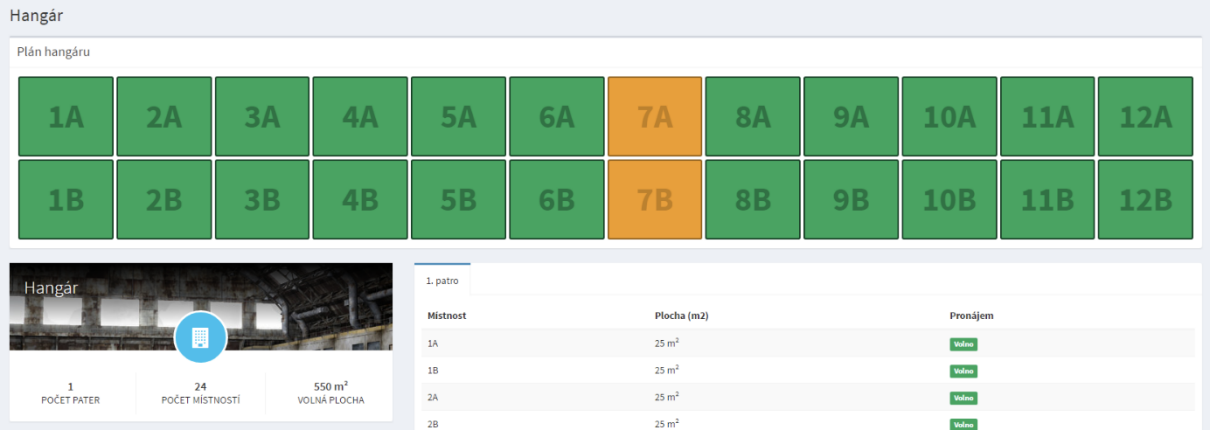
Pro vykreslování jednotlivých prvků se využívá několik SVG prvků, pro schematické znázornění velikosti objektu se používá <rect>, informace v něm jsou pak vykresleny pomocí <text>.

Aby se zamezilo přetékání dlouhých textů mimo objekt, je velikost textu dynamicky přepočítávána vzhledem k velikosti daného objektu.

Pro správné zobrazení v koordinačním systému SVG je využitý atribut viewbox, který pro relativní rozměry SVG kontejneru umožní nastavit velikost mřížky, tzn. pro kontejner o šířce 200 pixelů nebo o šířce 1 300 pixelů, je možné použít stejné hodnoty x a y, resp. width a height pro zachování správných poměrů objektů. Toto nastavení společně s atributem preserveAspectRatio none, zajistí responzivní chování zobrazení. [9]

4.3 Zobrazení hangárů

Vizualizace jednotlivých místností v budově typu hangár je řešena jinak než u klasických budov. U klasických budov se využívá technologie SVG, u hangárů se pak využívá pouze CSS a HTML elementů. Pro zobrazení hangárových místností je tento způsob efektivnější z pohledu zadávání jednotlivých hodnot.



Obrázek 21 - Detail budovy (hangár)

V horní části obrazovky se pak zobrazí schématický nákres plánu hangáru. Zelené kóje jsou volné, oranžové pak obsazené. Toto zobrazení můžeme vidět na Obrázku 21.

5 Implementace logiky aplikace

V této kapitole popisuji, jakým způsobem je vyvinutý backend aplikace, včetně popisu nejvíce používaných technologií.

Celý systém je naprogramovaný v jednom z nejrozšířenějších webových frameworků Symfony ve verzi 3.1. Tento Framework je úzce spjatý s architekturou MVC, která se pro tento typ projektu dobře hodí z důvodu oddělení logické vrstvy od vrstvy prezenční.

5.1 Přehled použitých technologií

V pozadí projektu je použito několik technologií starajících se o interakci s uživatelským rozhraním nebo o ukládání dat.

5.1.1 PHP + MySQL

PHP je populární skriptovací jazyk primárně určený k vývoji webových aplikací, ale v dnešní době užívaný v různých odvětvích IT. Společně s databází MySQL dnes tvoří nejčastější dvojici technologií pro webové stránky jako takové.

Na tomto projektu se počítá s využitím PHP veze 7 a vyšší, které jsou výrazně dále v oblasti bezpečnosti a mnohem rychlejší než předchozí verze, nicméně je zachována kompatibilita až do verze 5.5.9

5.1.2 Symfony

Symfony je MVC Framework již několik let vyvíjený jako open source projekt, do kterého už přispělo přes 1 400 vývojářů z celého světa, tím se tak stává jedním z nejrychleji se vyvíjejících projektů.

Jedná se o sadu několika balíčků, kde každý je určen pro řešení jiného problému, od řízení práv, přes přístup k databázi po vykreslení šablon.

5.2 Využívané knihovny

V této kapitole se věnuji popisu zajímavých knihoven, které jsou použity v rámci této aplikace.

5.2.1 Doctrine

Doctrine je další open source knihovna, která je úzce spjata se Symfony a některé její komponenty jsou integrovány přímo do frameworku. Je to knihovna zajišťující komunikaci s databází a následné další zpracování dat jak směrem do, tak i směrem z databáze.

5.2.2 ORM a entity manager

Jednou ze základních části Doctrine je ORM neboli objektově relační mapování dat. Ta se stará o mapování dat příchozích z persistentní vrstvy (v tomto případě MySQL) na instance objektů entit v systému, a také naopak řeší serializaci objektů ze systému do databáze. O ukládání a načítání dat z databáze se stará entity manager, ten uvnitř využívá návrhový vzor unit of work, který v sobě ukládá, co všechno je nutné na konci operace uložit do databáze.

5.2.3 Entita

Entita je třída, respektive její instance. Jedna entitní třída reprezentuje jednu tabulku v databázi, instance této třídy pak reprezentuje řádek v této tabulce. Implementace entity zajišťuje rozhraní pro manipulaci s jejími daty. Protože se jedná o objektově relační mapování, entita nemusí obsahovat pouze primitivní datové typy, ale může mít i referenční vazby na další entity, popřípadě jejich kolekce.

Pro správné mapování relací a také pro definování, jak se má daná property ukládat v databázi je v Doctrine několik způsobů zápisu.

Definice meta informací třídy

Na Obrázku 22, 23 a 24 lze vidět různé způsoby zápisu struktury tabulky „message“

```php
<?php
/**
 * @Entity
 * @Table(name="message")
 */
class Message
{
    //...
}
```

Obrázek 22 - Definice entity pomocí PHP anotace

```xml
<doctrine-mapping>
  <entity name="Message">
      <!-- ... -->
  </entity>
</doctrine-mapping>
```

Obrázek 23 - Definice entity pomocí XML

```yaml
Message:
type: entity
  # ...
```

Obrázek 24 - Definice entity pomocí YAML

Definice meta informací jednotlivých sloupečků tabulky

Takto je v Doctrine řešena konfigurace jednotlivých sloupečků tabulky

```php
<?php
/** @Entity */
class Message
{
    /** @Column(type="integer") */
    private $id;
    /** @Column(length=140) */
    private $text;
    /** @Column(type="datetime", name="posted_at") */
    private $postedAt;
}
```

Obrázek 25 - Definice sloupců tabulky pomocí PHP anotací

5.2.4 Schema builder a migrace

Protože jsou veškeré informace o struktuře databáze zapsány ve zdrojovém kódu aplikace, je možné automatizovat vytvoření, inicializaci i případné změny v databázi. Pro manipulace se strukturou databáze se používá Symfony Console. Doctrine implementuje několik příkazů, které slouží pro ovládání manipulace.

Doctrine:schema:create – skrz reflexi načte informace o jednotlivých entitách a pokud jsou tyto informace validní, vytvoří schéma databáze: tabulky, primární klíče, cizí klíče

Doctrine:schema:update – stejně jako create načte informace o entitách, dokáže však zjistit rozdíly oproti aktuálnímu stavu databáze a na základě těchto rozdílů databázi modifikovat

5.2.4.1 Migrace

Migrace je proces, který umožňuje modifikovat schéma databáze. Existují dva typy: dopředná migrace modifikuje původní databázi na novou verzi, zpětná migrace (tzv. rollback) migruje databázi na předchozí verzi. Migrace databáze je výhodná při úpravách struktury v databázi na již fungujících systémech.

V projektu je použita knihovna Doctrine Migrations, která přidává podporu pro automatizované vytváření migrací. Každá migrace je třída, která implementuje dvě metody up() a down(). Tyto metody se pak volají při spouštění jednotlivých migrací. Pro dopřednou migraci se volá up() a pro rollback down(). Pro práci s migracemi také existují příkazy do Symfony Console. Příklad třídy migrace lze vidět na Obrázku 26. [10]

Doctrine:migrations:diff – podobně jako schema:update načte rozdíly ve struktuře, neprovede ale modifikaci databáze, ale vytvoří třídu migrace, a tělo metod up() a down().

Doctrine:migrations:migrate – spouští instance jednotlivých migrací

```php
<?php

namespace Application\Migrations;

use Doctrine\DBAL\Migrations\AbstractMigration;
use Doctrine\DBAL\Schema\Schema;

/**
 * Auto-generated Migration: Please modify to your needs!
 */
class Version20170210224343 extends AbstractMigration
{
    /**
     * @param Schema $schema
   x  */
    public function up(Schema $schema)
    {
        // this up() migration is auto-generated, please modify it to your needs
        $this->abortIf($this->connection->getDatabasePlatform()->getName() != 'mysql',
'Migration can only be executed safely on \'mysql\'.');

        $this->addSql('ALTER TABLE vp_building ADD deleted_at DATETIME DEFAULT NULL');
        $this->addSql('ALTER TABLE vp_fixed_ground ADD deleted_at DATETIME DEFAULT NULL');
        $this->addSql('ALTER TABLE vp_fixed_place ADD deleted_at DATETIME DEFAULT NULL');
        $this->addSql('ALTER TABLE vp_floor ADD deleted_at DATETIME DEFAULT NULL');
        $this->addSql('ALTER TABLE vp_ground ADD deleted_at DATETIME DEFAULT NULL');
        $this->addSql('ALTER TABLE vp_object_entity ADD deleted_at DATETIME DEFAULT NULL');
        $this->addSql('ALTER TABLE vp_room ADD deleted_at DATETIME DEFAULT NULL');
    }

    /**
     * @param Schema $schema
     */
    public function down(Schema $schema)
    {
        // this down() migration is auto-generated, please modify it to your needs
        $this->abortIf($this->connection->getDatabasePlatform()->getName() != 'mysql',
'Migration can only be executed safely on \'mysql\'.');

        $this->addSql('ALTER TABLE vp_building DROP deleted_at');
        $this->addSql('ALTER TABLE vp_fixed_ground DROP deleted_at');
        $this->addSql('ALTER TABLE vp_fixed_place DROP deleted_at');
        $this->addSql('ALTER TABLE vp_floor DROP deleted_at');
        $this->addSql('ALTER TABLE vp_ground DROP deleted_at');
        $this->addSql('ALTER TABLE vp_object_entity DROP deleted_at');
        $this->addSql('ALTER TABLE vp_room DROP deleted_at');
    }
}
```

Obrázek 26 - Příklad třídy migrace

5.2.5 Nette/Utils

Jedná se o balíček z jiného MVC frameworku Nette. V této knihovně jsou třídy a metody umožňující například práci s textovými řetězci, asociativními poli, práci s obrázky nebo bezpečnější kódování/dekódování JSON.

5.2.6 Twig

Twig je výchozí šablonovací systém, který Symfony používá. Je navržen tak, aby vývojáře striktně tlačil k tomu, aby se veškerá logika odehrávala v modelové vrstvě a do view pak už přichází jen předpočítaná data, která se jednoduše zobrazí.

Syntaxe vychází z klasického HTML, je ovšem rozšířena o operátory umožňující psát rozhodující bloky nebo cykly pro procházení kolekcí. Výhodou je možnost pracovat s opakujícími se bloky šablon, popřípadě jednoduše do sebe tyto bloky zanořovat. Další výhodou je také využití dědění šablon, kterým pak lze jednoduše navrhnout layout pro informační systém, kdy je v hlavní šabloně definovaný prostor například pro menu nebo jiný prvek opakující se na všech podstránkách, ale o vykreslení obsahu dané podstránky už se postará šablona, která z této hlavní šablony dědí.

Velkou výhodou Twigu je možnost jednoduše přidávat vlastní funkce a filtry, pomocí filtrů je pak možno jednoduše sjednotit formátování času, případně čísel atd. [12]

5.2.7 Ukládání a manipulace s hesly

Jelikož je dnes kladen čím dál větší důraz na zabezpečení citlivých dat a dat obecně, tak i zde je kladen důraz na bezpečnost.

Po vytvoření nového účtu je nutné účet aktivovat na speciální URL doplněné o ověřovací token, na této URL je formulář pro přiřazení hesla k účtu. Po úspěšném dokončení procesu je uživatel aktivován a daná URL již není platná. Podobným způsobem probíhá i proces změny hesla, uživatel po kliknutí na „Zapomněli jste heslo“ vyplní svou e-mailovou adresu, na kterou mu vzápětí přijde zpráva s odkazem na zvolení nového hesla. Je tak zamezeno zasílání hesel v čitelné podobě po nezašifrovaných spojeních, jako je například SMTP.

Samotné heslo je pak v databázi uloženo v nečitelné (hashované) podobě pomocí algoritmu bcrypt s parametrem cost = 10, s přidáním tzv. salt řetězce, který by případnému útočníkovi znemožnil použití rainbow tabulek.

6 Uživatelské testování aplikace a zhodnocení výsledků

Tato kapitola popisuje, jakým způsobem byla aplikace postupně testována a upravována na základě požadavků z testování vzešlých.

Na úvod je nutno zmínit, že zadavatel už v minulých letech aplikaci řešil s jiným studentem, který projekt řešil technologií .NET, ale nebyl schopný zadání dokončit. Na základě jeho teoretického zpracování zadání a konzultace s vedoucí byla připravena první verze mé aplikace.

6.1 Průběh vývoje a úpravy zadání

Během vývoje proběhlo několik schůzek přímo na obecním úřadě Velké Polomi, ze kterých vyšlo několik změn, které byly v průběhu práce postupně zapracovávány do systému.

6.1.1 Úvodní schůzka

Společně s prezentací této nové verze aplikace, proběhla první společná schůzka na obecním úřadě ve Velké Polomi, které se zúčastnili paní starostka Ing. Bubeníková a místostarosta Ing. Chudoba za Velkou Polom, za VŠB potom já, společně s vedoucí této bakalářské práce Ing. Slaninovou a Bc. Moravcem, který na tomto projektu již dříve pracoval v rámci své bakalářské práce a nyní i práce diplomové.

Z této první schůzky vyplynulo mnoho změn zejména v oblasti práce se smlouvami a dodatky, ale obecně se také vydefinovalo, jakým způsobem se bude se systémem manipulovat a k čemu bude ve skutečnosti sloužit.

6.1.2 Druhá schůzka

Na druhé schůzce proběhla prezentace implementace požadavků z minulé schůzky. Zároveň se znovu upravil proces práce se smlouvami a dodatky, tato verze odpovídá aktuálnímu stavu aplikace. Dále byly sepsány body, které jsou nutné zahrnout do verze aplikace, která bude předána k prvnímu testování na straně klienta, dosud veškeré testování probíhalo na mé straně, případně ze strany Ing. Slaninové nebo Bc. Moravce.

6.2 Průběh testování

Projekt byl zprovozněn na cloudové službě Amazon Web Services, kde byl pro testování vytvořen server, se všemi potřebnými službami. Na tomto serveru pak probíhalo veškeré testovaní.

Od zapracování požadavků ze druhé schůzky byla aplikace předána ke klientskému testování, konkrétně panu Chudobovi, ze strany VŠB pak kromě mě aplikaci testoval také pan Moravec. Byla testována jak práce s uživatelským rozhraním, tak funkčnost celého systému a jeho logika.

6.3 Zpracování dodatečných požadavků

Z testování vzniklo několik dalších požadavků na drobnější úpravy, ale také několik složitějších funkcionalit např. „Hromadné přidávání objektů“. Tyto požadavky vyplynuly jak z další schůzky, které se zúčastnil pan Moravec, tak i z telefonické či e-mailové komunikace mezi mnou a p. Chudobou. Veškeré tyto požadavky byly zapracovány do aktuální podoby aplikace.

7 Návrh na další rozšíření aplikace

V této kapitole se věnuji možným rozšířením systému o další funkce, které by dále usnadnily správu smluv a další součásti systému.

Rozšíření vyhledávání relevantních prostor

Aktuálně lze vyhledat relevantní prostor pro nového nájemce pouze na základě zadaných rozměrů. Systém by bylo možno dále rozšířit o filtraci na základě služeb, které jsou v dané místnosti, případně celé budově možné.

Predikce ceny pronájmu

Na základě sbíraných dat, by bylo možné predikovat, jaká by měla být stanovena cena nájmu. Systém by pak mohl na základě dalších potřebných parametrů dopočítat reálnou konečnou cenu za nájem.

3D vizualizace

Aktuálně systém implementuje 2D vizualizaci plánů areálů a budov. Jelikož se ale jedná o reálné objekty, tak by se například při vizualizaci budovy dalo využít 3D modelů budov pro přesnější znázornění.

Automatické upozornění správci

Je pravděpodobné, že se se systémem nebude pracovat na denní bázi, ale bude využíván především na základě požadavků klientů o nové pronájmy, případně o úpravy stávajících smluv. V tomto modelu by tedy mohlo najít uplatnění automatické upozorňování správců, kteří by tak mohli být upozornění například pomocí e-mailu na končící smlouvy nebo dlouho nepronajaté objekty apod.

Analýza nájmů v čase

Na základě sesbíraných dat by bylo možno jednoduše sestavit graf, který by zobrazoval míru zaplnění jednotlivých objektů v čase. Dalo by se tak jednoduše zjistit v jakém období je pronajato nejvíce objektů zároveň, popřípadě jestli má počet nájmů rostoucí či klesající tendenci.

Přehled plateb klientů

Aktuálně se v systému zadávají platby za služby pouze jako predikce, není nijak řešeno, jestli jsou dané nájmy řádně placeny. Bylo by možné systém rozšířit o správu plateb nájemců, kde by byly všechny platby evidovány, včetně toho, jestli byly zaplaceny v řádném termínu apod. Z těchto dat by se dále mohla vypočítat například predikce cashflow, případně odhalit problémové nájemce.

8 Závěr

Cílem práce bylo vytvořit informační systém, umožňující správu pronajímaných objektů, který pomocí 2D vizualizace dokáže dobře informovat o zaplnění jednotlivých objektů a umožní tak například rychle a přehledně zjistit, které prostory je možné nabídnou potenciálnímu zájemci o pronájem. Mým záměrem bylo vytvořit kvalitní aplikaci, na kterou bude možné dále navázat a dále ji rozšiřovat.

V úvodu práce se věnuji popisu aktuálního stavu. Z tohoto popisu vyplývá, že aktuální způsob vedení správy nájmů je nedostačující, z čehož také vznikl požadavek na vytvoření tohoto informačního systému. Navazující část práce je pak věnována analýze požadavků a popisu objektového modelu, který na základě této analýzy vznikl.

Následující kapitoly se pak věnují samotnému systému a popisu použitých technologií jak pro uživatelské rozhraní, tak pro logiku aplikace jako takové, včetně zdůvodnění výběru SVG pro vytvoření 2D vizualizace.

V závěru práce pak zmiňuji, jakým způsobem probíhala komunikace s klientem a jak se projekt vyvíjel na základě výsledků uživatelského testování. V návaznosti na to jsem pak vypsal několik funkcionalit, o které by bylo možno systém rozšířit v další fázi.

Literatura (reference)

[1] A. Sellier, „Language Features | Less.js,“ [Online]. Available: http://lesscss.org/features/. [Přístup získán 1 6 2016].

[2] M. Otto, „Bootstrap - The world's most popular mobile-first and responsive front-end framework.,“ [Online]. Available: http://getbootstrap.com/. [Přístup získán 1 1 2014].

[3] D. Methvin, „jQuery API Documentation,“ 1 1 2015. [Online]. Available: http://api.jquery.com/.

[4] Microsoft, „Quick Start - TypeScript,“ [Online]. Available: https://www.typescriptlang.org/docs/tutorial.html. [Přístup získán 1 8 2016].

[5] J. Mrozek, „JavaScript na serveru: moduly a npm,“ 12 10 2012. [Online]. Available: https://www.zdrojak.cz/clanky/javascript-na-serveru-moduly-a-node-package-manager/. [Přístup získán 1 1 2015].

[6] M. Malý, „REST: architektura pro webové API,“ 3 8 2009. [Online]. Available: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/. [Přístup získán 10 8 2016].

[7] M. Michálek, „Frontendistův průvodce Gruntem,“ 4 9 2014. [Online]. Available: https://www.zdrojak.cz/clanky/frontendistuv-pruvodce-gruntem/. [Přístup získán 1 8 2016].

[8] M. Hassman, „Začínáme s HTML5 canvasem,“ 9 4 2009. [Online]. Available: https://www.zdrojak.cz/clanky/zaciname-z-html5-canvasem/. [Přístup získán 6 2 2017].

[9] [Online]. [Přístup získán 6 2 2017].

[10] J. Jenkov, „SVG Viewport and View Box,“ 16 1 2015. [Online]. Available: http://tutorials.jenkov.com/svg/svg-viewport-view-box.html.

[11] Doctrine Team, „2. Migration Classes — Doctrine 2 Migrations 2.2 documentation,“ [Online]. Available: http://docs.doctrine-project.org/projects/doctrine-migrations/en/latest/reference/migration_classes.html. [Přístup získán 11 2 2016].

[12] SensioLabs, „Documentation - Twig - The flexible, fast, and secure PHP template engine,“ [Online]. Available: https://twig.sensiolabs.org/doc/2.x/. [Přístup získán 1 1 2015].

Struktura přiloženého média

/vp-building-managment	Zdrojové kódy aplikace
/install.txt	Popis spuštění projektu